VIPER

VOLUME 2, ISSUE 10 AN ARESCO PUBLICATION JUNE 1980, $2.50

TABLE OF CONTENTS

SUBSCRIPTION RATES, ADVERTISING RATES, & OTHER INFORMATION

The VIPER is published ten times each year by ARESCO, at 6303 Golden Hook, Columbia MD, 21044, and mailed to subscribers on or around the 15th of each month except June and December. The single copy price is $2 per copy and the subscription price is $15/10 issues of the current volume. Subscriptions do not carry over from one volume to the next, and readers who want less than the full volume should send $2 for each issue desired. Renewals are accepted during the last two months of the current volume year, and the first issue of each volume is published in July.

Second class postage paid in Columbia MD 21045 (USPS 520-550).
Postmaster: Send all address changes to ARESCO, P O Box 1142, Columbia MD 21044.

Subscriptions: Subscription orders should be sent to P O Box 1142, Columbia, MD, 21044; not to the street address given above. USA residents: $20/10 issues mailed second class; $25/10 issues mailed first class. Non-USA residents: $20/10 issues mailed second class; $30/10 issues mailed first class. VISA/ MC/ personal checks, cash, money orders accepted as payment. All funds should be in US dollars. Checks drawn on foreign banks should include any exchange rate difference in currencies.

Advertising rates: Non-commercial ads by subscribers: $10/3 lines of 60 characters each. Non-commercial ads by non-subscribers: $15/3 lines of 60 characters each. For commercial ads, please write for advertising rates. Payment must accompany ad in any event.

Staff: Publisher: Terry L Laudereau, ARESCO, Inc.
Editor: Tom Swan
Subscriptions: Sandy Nolan

Contributions: Readers are encouraged to submit material of interest to VIP owners. However, it must be understood that the function of the VIPER is to duplicate the materials--and to mail them out to other VIPER readers. We cannot pay (sigh) for your efforts. In time, perhaps.....

EDITORIAL

The big old ciruela tree in front of our place is turning a young green, the leaves just barely covering its scraggy winter bareness. Black birds, I think they are mockingbirds, sail back and forth carrying nest building supplies. By the time we begin Volume 3 of the VIPER, that tree will be a dark green dabbed and dotted with yellow-red fruit. The baby birds will just be learning to fly, and to hide in the lush trees when the rains come. That will be a time to look ahead to the future, and ten more issues packed with VIP information, games, programs and a little more of our usual craziness. I'm looking forward to it. Hope you are too!

At about the same time, I expect Hayden Book Company to be bringing out my COSMAC 1802 programming manual. Rather than spill the beans before they were in the pot, I decided to delay any word about the project until I was sure it would be published. Naturally, I wanted all of you to be the first to know. It's a book for those who want to learn machine language programming but don't know where to start; plus it's a complete manual on 1802 programming concepts and features. There's a small subroutine library and a mini 1802 assembler/disassembler with which, by the way, I wrote and tested most of the code in PIPS FOR VIPS IV. Enough proud father boasting -- I'll let you know more as soon as they tell me.

Some expected additions for the next volume of VIPER include arrows, brackets and greater than/less than symbols for the typewriter. Scribbling these in by hand is such a chore, I sent off for the extra keys a few weeks ago. This may seem to be a trivial point, but if I forget to fill in a less than symbol by hand, the result makes no sense. Plus, the type will look better. While I'm on the subject, we all owe thanks to my wife Anne for typing every word in these pages. She's one of those behind-the-scenes people who deserve some recognition once and a while. Also, Sandy in ARESCO's offices -- i.e. my Mexican connection! And it's simply not possible to thank Rick and Terry enough for their unending dedication to this newsletter.

To authors, we look forward to seeing more of your great work. It's you, not we, who really deserve thanks from all of us. If you have the time to type your manuscript, we'd appreciate the effort, but there's nothing wrong with a "hand-u-script." Just try to write as clearly and as neatly as you can so we don't inadvertantly introduce any errors.

Of course, this message isn't goodbye! It's "Hasta Julio." See you then.

Tom Swan

TOM

Taxco, MEXICO

02.10.02

A SAD NOTE FROM THE EDITOR

Today it is April 15, 1980 and I have just been made aware of ARESCO's decision not to continue publishing the VIPER. Perhaps Terry will explain to you what has happened, I do not know, but the news is as much of a shock to me as I am sure it is to many of you.

Also today I learned of the erratic publishing schedule for the last two issues. My arrangement with ARESCO was to provide camera ready copy by the 15th of each month, a deadline I have been most careful to meet without fail. Because of the time lag between my work done here in Mexico where we live and ARESCO's offices in Maryland, I had no idea until now that you weren't receiving your issues on time. Even so, I wish to offer my apologies to all of you. Hopefully the fatter issues you did receive made up for the missing publications.

If we had continued, I think the VIPER would have entered its best period. Already we were receiving some fine articles and I feel the content of the newsletter was growing more professional all the time. Before I began to edit each issue, I was mostly concerned with my own material, though along with the rest of you, I devoured each issue as soon as it arrived. Afterwards, and from reading many of your comments, I began to see how the VIPER could better serve all of you. One thing I do agree that was wrong was to publish material needing the routines from PIPS to run. Seriously, we were not trying to coerce anyone to buy the books. The articles came so we published them. Still, you are correct, readers. Programs published here should run without having to buy a book from someone else. There's nothing wrong with writing books, and I plan to write some more. But if we had gone on to next year, I would have followed a policy that most of the material be for everyone. (By the way, 180 pages of PIPS IV was ready to be released when ARESCO made their announcement to me. The subject is Graphics and Animations for the VIP and most Elf owners will find routines that don't require the VIP operating system or hardware to run. If anyone is interested I suppose I'm in search of a publisher!)

That brings up another, and final point. Should anyone be interested in continuing the VIPER, I am willing to help -- either as editor or as a contributor. For that reason, I've decided to include my address below and any interested people may write to me directly.

To me, the VIP was just beginning to get off the ground. Some of you have probably seen RCA's new advertising campaign and new $99 version of the VIP. My 1802 manual will be coming from Hayden Books soon and I really hate to think we will all lose touch.

However, if this is goodbye, let me thank you all for your letters, your criticisms (they were all offered and received to be constructive in nature) and your compliments. I am especially sorry that much of the fine material sitting in the drawer will never be printed. And thank you for the opportunity to at least get my feet wet as your editor. It was good while it lasted and was an experience I will not soon forget.

Good luck to all of you.

Tom Swan
Apartado #38
Taxco, Gro., MEXICO

Dear VIPER - Recently, I presented a few ideas for expanding VIP TINY BASIC (VTB). Here are some more. You always lose two lines on the CRT with "READY" and the prompt. This is sort of a nuisance when writing and editing, having only five lines to start with. If your last PRINT statement is short (example: L# ? C), follow it with ;" "; and READY will appear on the last printed line (if there's room) instead of on the next line. This doesn't sound like much, but it cuts re-listing to get back the 4th line up from the bottom by about 50%.

My first program that ran more than 2K calculates the price of gold (timely, huh?). I whittled it down to fit. I'm strictly a non-math type, and I amazed myself by even being able to work out the formula. This is my first program so far that actually computes anything. Of course double precision floating point would be a lot better.

Now that my computer will compute and do all those other wonderful things, I could use a printer. Apparently two or three VIPs have printers connected. But they all seem to use 1802 machine code. I have found a pseudo-LPRINT that works with VTB and a strobe, too. It also make a nifty DEC-HEX converter if you have a HEX display on the output port. I'm working out the details of a printer conversion, but I don't have a printer yet. They are all too expen$ive. I'm working on an idea that uses the old W.U. Deskfax units (at surplus fleamarkets for $15 or $20) to build a printer/plotter. Some mechanical changes will be necessary. A UART could serialize VIP's parallel output. The fax is a serial input device using a charged wire stylus to mark electro-sensitive paper. You move the paper and/or the stylus and zap the paper to make a mark. Timing does the rest. I have 3 or 4 extra fax units and about 27,000 sheets of paper for them. Roll paper can be used in other machines. Fax paper is not the same as thermal paper, but that may work too. Alphanumerics and graphics in any format can have much better resolution with this than VIP's 1861 TV interface. Would anyone care to try this? My address is included for those who do. - Don L. Hartley Rt. A Box 168D, Yellville, Ark. 72687

Dear VIPER - My trek through your PIPS FOR VIPS packages continues. I am one of those ELF owners who is keying in all those pages of code. As a result, I am likely to run into any bugs in your object code. Here are the latest.

In VIP-OKER (PIPS III) on page 57 the CHIP-8 instruction at 05DE should be changed from ABC0 to A8C0 which will correctly point to the "FOLDS" character string.

Also in this same program, I found and bypassed a questionable piece of code on page 87. The code at 0C9C reads: 01A4 F1F0. Please explain what these two CHIP-8 instructions do. I changed them to 8100 1CA0 (V1=V0, GOTO CA0) and the program works fine.

Also in PIPS I, there is a 2 page Editor machine language program used to display CHIP-8 code. Could you verify the object code against a version you know works? Thanks! - Chuck Reid

Dear Chuck - Thank you for the corrections. As many times as I have typed or written B's for 8's and vice versa, I'm ready to go back to octal or use the original hex digits U,V,W,X,Y, and Z! The MLS you bypassed takes the place of the BMMM instruction in the CHIP-8 Interpreter. It's not needed for bets less than $10. The routine serves to combine V1 into VØ allowing successive decimal keypresses to go into a single variable. Here's the little MLS you skipped.

```
01A4  E6 45 A6 45 A7 06 FE 56 FE FE F4 E7
01B0  F4 56 D4 -- -- -- -- -- -- -- -- --
```

This is relocatable (i.e. uses no branches) so it may go anywhere. The FXFY following the call to the sub serves to identify X and Y as the active CHIP-8 variables.

As far as I know the object code for the editor is correct as printed. It does make use of VIP operating system routines, however, which may not be available on your ELF. Are there any ELF owners out there who know the answer? Thanks for writing. - Tom

Dear VIPER - A few months ago I purchased my VIP with color and Simple Sound. I now have TINY BASIC and the VP-611 keyboard. I have also gone back to the dealer for volume 1 of the VIPER along with the tape and info "PIPS FOR VIPS" by Tom Swan.

I have just completed a modification to the Text Editor-21 program that allows my keyboard to work with this program. As requested in the introduction to "PIPS FOR VIPS," here are the modifications.

```
28 to 37 Wait for
29    2C keypress
2A    30
2B    28
2C    6B Input to
2D    B8 R8.1
2E    F8
2F    04
30    A8 Key
31    88 debounce
32    3A
33    31
34    F8

39 to 98 Get 8.1
3A    FA And with 7F
3B    7F Mask off MSB
3C    B8 Put in R8.1
3D    FF Compare to
3E    20 20
3F    33 Jump to B3
40    B3 if more than
         20
41    F8 Initialize RC
42    81 for hex
43    BC Keyboard scan
44    F8

B4 to 03 for ASCII
B5    9F character
B6    D4
B7    02
B8    C6
B9    30
BA    28
C0    98 Get R8.1
C1    FA Allow only
C2    07 08-0F codes
C3    F9
C4    08
C5    B8 Do function
```

```
35   04          45   95              C6   D4 routine
36   A8          46   AC              C7   02
37   37          47   30 Jump to 00C0 C8   00
38   34          48   C0              C9   30 Loop back for
                 B3   D4 Do routine   CA   28 next key
```

With these changes control "L" functions as key "C" and the second key is selected on the hex keyboard as before. With the control key and keys "H through O" the rest of the functions are available. - R. Winterton

Dear VIPER - Although late in responding, I hope you can still use the information. I would buy a light pen, and I'm interested in the "stringy floppy." Right now my ELF II is not a VIP, but soon it will be. I really like the VIPER - especially Tom Swan's articles. - Kendall Stambaugh

EDITOR'S NOTE: Thanks for writing, Kendall. To other readers who have expressed an interest in the light pen, look for applications notes in next year's volume of VIPER. I've got mine working -- soon as I get a program together, I plan to write it up. - Tom

Dear VIPER - I discovered an interesting and baffling bug in my register and breakpoint program (VIPER, Oct. 1978) that might also confuse its users. It hadn't shown up until now, since it appears only under rather special circumstances.

Essentially, the system will break in rare instances at unspecified places in the program. The circumstances are these:

1. An IDL (00) exists in the program at some location P.
2. A branch to location P+1 has just been executed.
3. An interrupt (caused by the 1861 TV interface) has occurred just before the instruction at location P+1 is fetched.

There is no way to prevent this kind of erroneous break from happening, hence I have no fix to offer for the program. The system depends on hardware interrupts (generated by the TV interface) as a means of capturing all the registers. These interrupts occur at regular (but long) intervals. When one occurs, the interrupt routine inspects the location R(P)-1. If that is an IDL, the program concludes the interrupt was caused by a breakpoint (another use of IDL). Usually that assumption is correct -- an IDL waits for an interrupt or DMA. A DMA is prevented from occurring by special instructions in the hardware breakpoint program, so an interrupt only causes a breakpoint.

However, a branch to an instruction just following an IDL can also look like a break, since the instruction preceding it is 00. Such a break will only occur if the interrupt happens to fall just after the branch, after R(P) has been changed, but before the next instruction is fetched. There is no way to determine whether such a break is valid or not after the fact.

The bright side of all this is that the breakpoint system won't miss a planted breakpoint, if the rules are followed. It just might come up with a breakpoint that you didn't set. - Bill Barrett

Dear VIP Owners,

I am negotiating with Terry to continue publication of VIPER uninterupted if at all possible. We both consider it of utmost importance to VIP users that our only means of information exchange be kept alive and serving our needs. The "big" computer hobby magazines rarely if ever acknowlege our existence. I intend to keep VIPER almost exactly the same for now. Later some changes and/or improvements will be considered.

This is your magazine-newsletter and I will depend heavily on your support and suggestions to make this effort successful. Will you help me make a smooth transition by renewing now for one year at the usual rate of $15 for 10 issues? Please add $10 for airmail out of USA 48. Your checks and money orders will be kept in a special account until I receive 500 renewals and/or new subs. I must have at least 85% of present subscriber response by June first to make the decision to print a July issue(vol.3,#1) and 100% will ensure a full year of publication. I want to get that issue to the printer on June 15 and in the mail by June 25 so you will get the issue early in July. If I do not get sufficient response all checks will be returned. Please include a stamped addressed envelope(sase) with your check. It costs about $80.00 to return 424 checks and a sase will insure immediate return if response is not sufficient. I sincerely hope I can return all your sase's unused with the first issue.

Looking to the near future, I expect to see more interest in VIPs now that RCA has announced a price of $99 for essentially the same system that most of us paid $ 275 for. (ouch!) The announcement of VP-701 full BASIC with floating point is very exciting and lends credence to my opinion that VIP owners can do anything that the "other guys" are doing and for a lot less money.

As your new publisher, I have several hints, tricks, and ideas to offer. Examples include a really cheap printer & interface to Tiny Basic, some VTB programming ideas, & a portable carry case/cabinet for your VIP that also holds a bunch of accessories and I/O expansion. I will contribute these and other ideas as time permits. Initially I will be counting on you for more of the fine articles and letters that have made VIPER an indispensable part of VIP-ing!

All material must be camera ready to be printed, as I have no text editing to begin with(soon I hope). I will use your sub. money only for getting out the first few month's issues until this venture can support itself and pay for future refinements. A larger subscriber base will enable and hasten improvements, and forestall a rate increase.

I will enjoy working with Terry and Rick during the startup phase. They did a fine job getting our newsletter started and I will be totally indebted to them and their support.

I welcome your comments. Address correspondence to: Don Hartley, Hartron, Rt. A Box 168D, Yellville, Ar. 72687.

Sincerely,

Don

Dear VIPER - I have found PIPS FOR VIPS volume 2 puzzling. I am interested in using the revised Editor program for an ASCII keyboard. Does the PIPS FOR VIPS tape contain the edit program?

Also, I found it helpful as a precaution to break the tabs on the back of cassettes. This prevents recording anything over the programs. - Marvin J. Bleiberg

Dear Marvin - The editor you mention is included on volume 1 of the PIPS series. However, any text editor could be used to prepare source code for Volume II's CHIP-8 Assembler. The code must follow the format explained in the book. Thanks for the tip on the cassettes. This is very important -- especially when using the assembler which requires a lot of tape handling. - Tom

1802 MEMORY DISPLAY PROGRAM

by Gerald Strope

I wrote this program to display the contents of memory, mainly to verify newly entered programs. This is very important in a microprocessor without memory protect, because one wrong instruction can cause many other memory positions to be altered, causing the program to enter self-destruct mode. The Cosmac VIP has the capability to display memory one byte at a time, but this is rather tedious when verifying up to 2K of memory. This program was developed on a 2K Cosmac VIP, but with a few modifications will run on the ELF-2.

Here are some of its strong points: a) It is contained in one page of memory (256 bytes) b) It runs alone in machine language independent of any other program such as CHIP-8. c) It can be loaded in any page of memory that is available.

There are three display options available:

Option 1 - 2 byte address - 2 bytes of data
Option 2 - 1 byte address - 3 bytes of data
Option 3 - no address - 4 bytes of data

Option 1 is the most convenient for displaying CHIP-8 programs because instructions are written in a two byte format. Option 2 allows the right half only of the address to display, and displays 3 bytes of data to fit more memory positions on the screen. Option 3 allows maximum data to be displayed. I suggest you change address AF to hex 'C0' to display four lines instead of five. With this change exactly 16 bytes of data are displayed, so every time the display is advanced the first byte displayed will be 00-10-20-30 etc. This option is good for copying down a program after it is debugged.

The chart in figure 1 shows the control byte changes for the different options. To use the program, perform the following steps:

1. Load the display program into the desired page. (The code is set to run at 0600 as printed.)

2. Alter memory locations 0000-0003 to C0 XX 00 (XX = page address where the display program is located. Normally XX=06.

3. Use the display program (C key will cause the memory location to advance).

4. When finished alter 0000-0003 back to original data.

Here are some additional notes on the use of this program. Watch out when using it in page 06 of memory. CHIP-8 uses the next to last page as a work area, so if CHIP-8 is executed the display program will be altered. Also the C key advances the display only because that was the last character set in the keyboard latch by

the operating system. If you enter this program from another editor or some other means, the last key you used before entering this program will be the advance key. I intentionally did not set any specific character into the keyboard latch to save program space.

You may wonder what that funny line is at the bottom of the screen. I didn't have room for the interrupt stack in one page of memory, so I used some of the display memory to hold it. It is quite instructive to see the stack in operation. I didn't realize it worked quite that way until I saw it in operation.

I hope you find this program as useful as I have.

MODIFICATIONS FOR ELF-2 USERS

The program uses the character patterns stored in the Cosmac VIP ROM. Figure 2 is a listing of this pattern. To run on the ELF-2, store this pattern using the same low order address. Set the high order address to the desired page. (For example: store the display program in page 05 and the pattern in page 06) One word of caution here, page 07 is set up as refresh memory. Do not try to load the pattern in the same page as refresh memory. Refer to figure 1 for information to relocate the refresh page. Also you must change the following address locations.

Address	from	to
1D	81	XX (XX=page bit pattern is in)
B2	3E	3F (allows input key to advance program)

Control Byte Chart - Figure #1

Address Location	2 data byte format	3 data byte format	4 data byte format
9F	7E	85	85
B1	D9	7E	85
C2	D9	7E	85

Address Location
AF=F0-Display 5 lines
AF=C0-Display 4 lines
08=XX (XX=page program is loaded in)
C9=XX (XX=page program is loaded in)
0D=XX (XX=page program will start displaying at)
19
2D=XX (XX=page to be used as refresh memory)
B5

Character Bit Patterns - Figure #2

10	F080F080	20	F080F010	30	F0909090
14	F0808080	24	F080F090	34	F0101010
18	F0507050	28	F090F010	38	10602020
1C	F0505050	2C	F010F090	3C	2070A0A0
				40	F02020

Register Allocation

R0 - Refresh
R1 - Interrupt
R2 - Stack
R3 - Main
R4 - Display subroutine call
R5 - Memory being displayed location
R6 - Display location
R7 - Pointer to bit pattern
R8 - Work register 8.0-counter, 8.1-character
R9 - Work register 9.0-counter, 9.1-character
RA - Left half-right half flip-flop
RB - Counter-1st,2nd,3rd,4th decision block
RC - Counter-divider bar positioning

Program Listing

Address	Code	Comments
00	F800AA	RA.0,R5.0,R6.0
03	A5A6ABBB	RB.0,RB.1=00
07	F806B1	R1.1,R3.1,R4.1=06
0A	B3B4	
0C	F806B5	R5.1=06=page to be displayed
0F	F825A1	R1.0=25=interrupt
12	F8FFA2	R2.0=FF=stack
15	F842A4	R4.0=42=display subroutine
18	F807B2B6	R2.1,R6.1=07
1C	F881B7	R7.1=81=display pattern pointer
1F	F87AA3	R3.0=7A=main
22	D3	P=R3
23	7270	Restore D,R2+1,restore X,P,R2+1
25	2278	R2-1 save X,P in M at R2
27	2252	R2-1 save D in M at R2
29	C4C4C4	No-op
2C	F807B0	R0.1=07 0700=refresh location
2F	F800A0	R0.0=00
32	80E2	D=R0.0,X=2
34	E220A0	R0-1,R0.0=D
37	E220A0	" "
3A	E220A0	" "
3D	3C32	Go to refresh if EF1=0
3F	3023	Go to return
41	D3	P=R3
42	A7	R7.0=D (start of display subroutine)
43	F805A9	R9.0=05 (character byte counter)
46	4756	M(R6)=M(R7) (character pattern-refresh)
48	8CFB02	Get RC.0, ex OR with 02
4B	3250	Go to 50 if D=0
4D	305400	Go to 54
50	06F90656	D=M(R6),or with M(R6)=06 (put bar between address and data)
54	F808A8	R8.0=08 (refresh advance counter)
57	281688	R8-1,R6+1,D=R8.0

Address	Code	Comments
5A	3A57	Go to 57 if D not 0
5C	2989	R9-1,D=R9.0
5E	3A46	Go to 46 if D not 0
60	86FA0F	D=R6.0,and with 0F (mask off left half)
63	FB0F	Ex or imm with 0F (check if display at right side)
65	3A6D	Go to 6D if D not 0
67	16	R6+1 (new line)
68	F8C3A3	R3.0=C3 (return location for main)
6B	3041	Go to 41 (return to main)
6D	F827A9	R9.0=27 (counter to back up R6 to next character location)
70	292689	R9-1,R6-1,D=R9.0
73	3A70	Go to 70 if D not 0
75	F8C3A3	R3.0=C3 (return location for main)
78	3041	Go to 41 (return to main)
7A	E269	X=2 turn on TV (start of main)
7C	30B4	Go to B4 (erase routine)
7E	F800AC	RC.0=00
81	85B8	R8.1=R5.0 (put low order M address in work register)
83	3087	Go to 87
85	45B8	R8.1=M(R5) (put M(R5) in work register)
87	FAF0	And with F0 (mask off right half)
89	F6F6F6F6	Shift right 4 times
8D	B9	Put in R9.1
8E	1A1C	RA+1,RC+1
90	30C8	Go to C8 (compare subroutine)
92	98FA0F	Get R8.1, and with 0F (mask out left half)
95	B92A1C	R9.1=D,RA-1,RC+1
98	30C8	Go to C8
9A	1B8B	RB+1,D=RB.0
9C	FB01	Imm or 01
9E	327E	Go to 7E if D=0
A0	8BFB02	D=RB.0,imm or 02
A3	3285	Go to 85 if D=0
A5	8BFB03	D=RB.0,imm or 03
A8	3285	Go to 85 if D=0
AA	F800AB	RB.0=00 (reset B.0 to 00)
AD	86FBF0	D=R6.0,imm or F0 (check for bottom of screen)
B0	3AD9	Go to D9 if D not 0
B2	3EB2	Go to B2 if EF3 not 1 (wait for key)
B4	F807B6	R6.1=07
B7	F8FFA6	R6.0=FF
BA	E6	R6=X
BB	F80073	Load 00, store and decrement
BE	863ABA	D=6.0,go to BA if D not 0
C1	30D9	Go to D9
C3	8A3A92	D=A.0,go to 92 if D not 0
C6	309A	Go to 9A
C8	F806BD	RD=06E0
CB	F8E0AD	" "
CE	99EDF3	D=9.1,X=D,M(RD)ex or D
D1	1D32D7	RD+1 go to D7 is D=0 (equal compare)

Address	Code	Comments
D4	1D30CE	RD+1 go to CE (compare again)
D7	4DD4	D=M(RD),P=R4
D9	F802AC	RC.0=02
DC	95B8	R8.1=R5.1
DE	3087	Go to 87
E0	0030	
E2	0139	Compare Table
E4	0222	1st byte=compare
E6	032A	2nd byte=pointer address of bit pattern
E8	043E	(example-the bit pattern for 0 is located
EA	0520	at address 30)
EC	0624	
EE	0734	
F0	0826	
F2	0928	
F4	0A2E	
F6	0B18	
F8	0C14	
FA	0D1C	
FC	0E10	
FE	0F12	

NOTE: this listing has the following options set in it. If you desire other options refer to figure #1.

1. 2 address bytes - 2 data bytes format
2. To be loaded in page 06
3. Program will start displaying at 0600
4. Display height = 5 lines
5. Refresh memory = page 07

NON-COMMERCIAL AD CORNER

Joseph F. Oberhauser has a slew (I can think of no other word) of VIP equipment for sale. VIP, cassette, printer, keyboard and more. Send inquiries to us and we'll pass them on to Joe.

This brings up an important point as I suspect Joe would not mind receiving direct replies. But if you want us to print your address in the newsletter, you must specifically give ARESCO and/or the VIPER permission to do so. We have been strict about this policy out of respect for those who do not have the time, energy (or sometimes the money) to reply to all the mail that can result from publishing their address. Only if you write "you have my permission to publish my address" or words similar to those will we print your address in the newsletter.
Thanks - Tom

LITTLE LOOPS by Tom Swan

STACK IT TO ME

CHIP-8 is stackless. Yes, there is a stack for CHIP-8 but it is not accessible to your programs. The CHIP-8 stack is used by the interpreter to store intermediate values needed later and to keep track of the paths, channels, twists and turns of your subroutine structure. When all subroutines RETURN, the stack provides a map of the way back. You do not have to be aware of the stack or how it operates. All is handled invisibly by the genius of the CHIP-8 interpreter.

As you become more interested in programming concepts, the stack is a DATA STRUCTURE you must learn inside out. Even if you have some stack handling experience, especially if your stack use has been limited to 1802 machine language, the following comments may be of use to you. Unfortunately, the stack procedures of the 1802 are unnecessarily confusing, and do not inspire correct stack handling. The comments from a reader in San Diego (who asked not to be identified) are quite correct. There should be a Pre decrement for the STX resulting in a DSTX rather than STXD. Good point.

A stack is a very simple structure to understand. The term LIFO for Last In First Out has been given to this common programming device as a description of the action of a stack. Have you ever played that children's game where everyone piles their hands on top of everyone else's? That's a stack, except that bringing new hands from the bottom of the pile would be illegal in a computer stack. If no one cheats, the only way to get to the fourth hand is to remove one-by-one, all the hands above it. The last hand on the pile (in the stack) must be the first one to be removed.

The sample program, written in CHIP-8, is intended to demonstrate stack input and output. When an item is inserted into the stack, programmers say the item is "PUSHed onto the stack." Taking an item off the stack is jargonized into "POPing the stack." PUSH and POP should become familiar terms to you. They are not descriptions of the sounds made by a certain breakfast cereal.

The two subroutines PUSH and POP may be used in your own programs to implement a CHIP-8 stack. Happily, no jumps were needed by either routine, so they may be relocated without rewriting any of the instructions.

Before calling either subroutine, you need to initialize two variables somewhere in your program. VA must be initialized to Ø and VB to the maximum number of items you will allow in the stack. VB may be any value from Ø to FF but will usually be set to 10 or 20 (hex) for a stack depth of 16 or 32 bytes. VB is set to 0C (12 decimal) in the sample program to avoid printing too many numbers on one line.

CALLING PUSH: Now that you have initialized the stack, you may PUSH a value onto the stack by first setting VØ equal to that value then

executing 2XXX where the X's are replaced with the address of the PUSH sub. Immediately on returning from the sub, check the value of VF. If VF=Ø, then VØ was successfully PUSHed onto the stack. If VF=1 on return, then the stack was full and VØ was not PUSHed. VØ is not changed. Note that this overflow condition does not prevent you from taking items off the stack to open up some room. Some textbooks treat overflow as a fatal condition -- that is, the program stops. This does not have to be the case with these CHIP-8 subs. The stack pointer will always be correct.

CALLING POP: To remove the topmost item from the stack, simply execute 2XXX with the X's set to the address of the POP sub. If VF=Ø on return, then VØ was set to the value of the top item of the stack. If VF=1, however, then the stack was already empty. Again, your program may continue without concern for this underflow condition. It simply indicates an empty stack, not a damaged one.

When an item is POPed from the stack, you should consider that item to be gone from the stack even though the value may still be in memory at I+VA+1. The sample program takes advantage of this fact to print out the stack -- it just saves the stack pointer, pops the stack to underflow while printing each digit, then restores VA. Normally, this is not good stack handling, however.

In the sample program, 0600 is used as the base address for the stack. This may be changed by setting the I pointer at 0228 and 0238 to a different address. The base address, in this case 0600, will never be used to store values in the stack. In other words, the stack always contains a wasted byte at the bottom. It is important that you do not modify the subs to avoid this insignificant waste. In a program that uses several stacks, this byte may be used to store the stack pointer for that stack. Thus any number of stacks may coexist in memory using only a single stack pointer CHIP-8 variable. Each stack would contain its own stack pointer always located at the base of the stack.

Because of the way "I" may be indexed,* the CHIP-8 stack will start at BASE+1 and grow to BASE+2, BASE+3...,+VB before overflow occurs. Most processors run the stack the other way, subtracting one from the BASE address before PUSHing. (Except for the 1802 which PUSHes, then subtracts one. -- Oh dear!) Don't forget which direction your stack is headed!

Finally, VB may be freed for other uses by using a permanent maximum value for the stack depth. The instruction at 0222 could be changed to: 4AXX where XX= the number of items allowed in the stack. For a program with a single stack, this would be a useful modification. For a multi-stack program, perhaps with memory assigned dynamically as needed, VB will be needed to control stack sizes.

Like textbook Chess openings, these subroutines represent the way to implement a software stack. You do not have to be aware of the stack pointer -- it will take care of itself. Stacks have uses in arithmetic expression processing, games, graphics and even interpreter design. We'll look into these possibilities in future issues. Until July, have fun!

*Anyone who tries to index "Me" is in big trouble, though.

PROJECTS:

1) The children's hand game mentioned earlier forbids taking hands from the bottom of the pile. If that rule is left in, what data structure is then represented by the pile of hands? Try to implement that structure in CHIP-8.

2) What's a simple way to use an entire memory page for the stack?

3) The sample program contains a skeleton of a CHIP-8 character set display. Unfortunately, as presented, the idea is rather clumsy. Can you think of a better character set display using basically the same idea, but avoiding conflicts with the I pointer? (The answer is contemplated as a possible article for a later issue and response is welcome.) Could a stack be used?

STACK SAMPLE PROGRAM

-VARIABLE ASSIGNMENT-

VØ - General parameter passing
V1 - General -- used in ONECHR sub and Main Loop
V2 - Next character index for printing strings (words)
V3-V9 - Not used
VA - Stack pointer. VA=Ø=empty/VA=VB=full
VB - Maximum depth (# items) allowed in stack
VC - X coordinate for display
VD - Y coordinate for display
VE - Holds VA during output of stack
VF - Flag variable. General use

LANGUAGE: CHIP-8 as printed in the VIP programming manual. Load into 0000-01FF

NOTES ON LOADING: Remember to load the MLS at 0400. This is included here, but was documented in last month's column.

When you are finished loading, record five pages from 0000-04FF and flip to run.

USING: Press any key less than hex A to PUSH that value onto the stack. Push any key greater than 9 to POP one value from the stack. After each operation, the stack is shown with the top of the stack to the left. Note that underflow occurs on POPing an <u>empty</u> stack, not on removing the last value.

```
      ;MAIN LOOP

0200  BEGIN: 6A00  ;VA=Ø  -- Initialize stack pointer to zero
  02         6B0C  ;VB=C  -- Initialize maximum stack size
  04  S1:    F00A  ;KEY   -- Get hexpad input in VØ
  06         6109  ;V1=9  -- Test if VØ is greater than
  08         8105  ;V1-V0 --   9 by subtracting from 9
```

```
020A          3F00  ;V1<VØ -- If negative result, then VØ > 9
  0C          1216   S2    -- Else VØ < =9/GO INSERT
  0E          2232   POP   -- POP an item off stack into VØ
0210          3F00  ;SK OK -- If flag not set, then skip
  12          22AC   PUNDR -- Else print underflow message
  14          121C   S3    -- Skip the next section always
  16  S2:     2220   PUSH  -- PUSH VØ onto stack
  18          3F00  ;SK OK -- If flag not set, then skip
  1A          2298   POVER -- Else print overflow message
  1C  S3:     22EA   STACK -- Show contents of stack
  1E          1204   S1    -- Go do it again

     ;END MAIN LOOP

          CHIP-8 SUBROUTINE -- PUSH VØ ONTO STACK

0220  PUSH:   6F01  ;VF=1  -- Preset flag to indicate possible overflow
  22          9AB0  ;SK ≠  -- If VA≠VB then skip.  No overflow this
                            --   time
  24          00EE  ;RET   -- Return with flag set=1.  Stack is full
  26          7A01  ;VA+1  -- Add one to stack pointer preparing to
                            --   insert
  28          A600   BASE  -- Set I to the base address of the stack,
  2A          FA1E  ;I+VA  --   then add VA to find the top
  2C          F055  ;PUSH  -- Insert VØ value into stack at X(VA)
  2E          6F00  ;VF=Ø  -- Set flag=Ø marking successful PUSH
0230          00EE  ;RET   -- Return from subroutine

          CHIP-8 SUBROUTINE -- POP STACK INTO VØ

  32  POP:    6F01  ;VF=1  -- Preset flag to indicate possible underflow
  34          4A00  ;SK≠0  -- Skip as long as VA≠0.  Stack is not empty
  36          00EE  ;RET   -- Return with flag set=1.  Stack is empty
  38          A600   BASE  -- Set I to the base address of the stack,
  3A          FA1E  ;I+VA  --   then add VA to find the top
  3C          F065  ;POP   -- Retrieve the value from the stack.
                            --   Put into VØ
  3E          7AFF  ;VA-1  -- Subtract one from the stack pointer
0240          6F00  ;VF=Ø  -- Set flag=Ø marking successful POP
  42          00EE  ;RET   -- Return from subroutine

                  SCROLL CONTROLLER

  44  SCRLC:  6F08  ;VF=8   -- See last month for scrolling details.
  46  SCRL1:  0400  ;SCROL  -- This sub needs the SCROLL MLS @ 0400
  48          7FFF  ;VF=1
  4A          3F00  ;SK=0
  4C          1246   SCRL1
  4E          00EE  ;RET
```

ONECHR SUBROUTINE

```
0250  ONECH: 70FF  ;V0-1  -- Adjust VØ. (1st character at index=Ø)
  52         810E  ;SHL   -- Shift VØ left to multiply*2. Save in V1
  54         801E  ;SHL   -- Shift V1 left to multiply*2. Save (*4)
                   --   in V0
  56         8014  ;ADD   -- Add V0+V1 for V0*6 total.  Save in V0
  58         A262   CHTAB -- Set I to base address of character table
  5A         F01E  ;I+V0  -- Add V0 to I to find right character
  5C         DCD5  ;SHOW  -- Print the character with code VØ
  5E         7C05  ;VC+5  -- Advance X coordinate for next printing
0260         00EE  ;RET   -- Return from subroutine

0262  CHTAB: F0 50 50 50 F0 00 -- Index 1/Char D
  68         F0 80 E0 80 F0 00 --   "   2/  "  E
  6E         90 D0 B0 90 90 00 --   "   3/  "  N
  74         F0 90 90 90 F0 00 --   "   4/  "  O
  7A         F0 90 F0 A0 90 00 --   "   5/  "  R
  80         90 90 90 90 F0 00 --   "   6/  "  U
  86         90 90 90 F0 60 00 --   "   7/  "  V
  8C         F0 80 E0 80 80 00 --   "   8/  "  F
  92         80 80 80 80 F0 00 --   "   9/  "  L
```

PRINT "OVER"

```
0298  POVER: 22E2   ILINE -- Reset VC,VD and scroll up
  9A  POVE1: 6200  ;V2=Ø  -- Set V2=Ø for indexing string
  9C         A2D2   OVER  -- Set I to first byte of string
  9E         F21E  ;I+V2  -- Add index (V2) to I
02A0         F065  ;GET   -- Get a byte of the string
  A2         4000  ;SK≠0  -- If not=Ø (end of string), then skip
  A4         12C0   PFLO  -- Else go print "FLO"
  A6         2250   ONECH -- Do sub to print one character
  A8         7201  ;V2+1  -- Add one to V2 index
  AA         129C   POVE1 -- Go loop until done
```

PRINT "UNDER"

```
  AC  PUNDR: 22E2   ILINE -- (See notes for POVER sub)
  AE  PUND1: 6200  ;V2=00 --
02B0         A2D8   UNDER --
  B2         F21E  ;I+V2  --
  B4         F065  ;GET   --
  B6         4000  ;SK≠0  --
  B8         12C0   PFLO  --
  BA         2250   ONECH --
  BC         7201  ;V2+1  --
  BE         12B0   PUND1 --
```

PRINT "FLO"

```
02C0  PFLO:  6200  ;V2=Ø  -- (See notes for POVER sub)
  C2  PFLO1: A2DE   FLO   --
  C4         F21E  ;I+V2  --
  C6         F065  ;GET
```

```
02C8          4000  ;SK≠0  --
  CA          00EE  ;RETN  -- PUNDER, POVER, and PFLO return here
  CC          2250   ONECH --
  CE          7201  ;V2+1  --
02D0          12C2   PFL01 --

                          CHAR STRINGS

02D2  OVER:   0407 0205 0000 -- Uses a special non-ASCII
  D8  UNDER:  0603 0102 0500 --   code for this program only
  DE  FLO:    0809 0400      --

                          INIT LINE

02E2  ILINE:  2244   SCRLC -- Do sub.  Scroll up one line
  E4          6C00  ;VC=Ø  -- Set X coordinate=Ø (left)
  E6          6D38  ;VD=38 -- Set Y coordinate=38 (bottom)
  E8          00EE  ;RETN  -- Return from subroutine

                          SHOW STACK

  EA  STACK:  22E2   ILINE -- Reset VC,VD and scroll up
  EC          8EA0  ;VE=VA -- Save stack pointer in VE
  EE          2232   POP   -- Pop an item from the stack
02F0          3F00  ;SK OK -- If flag not set, then skip
  F2          12FC   STAC2 -- Else done.  Go reset and exit.

      ;SHOW ONE NUMBER

  F4          F029  ;SET I -- I addresses ROM bits for VØ value
  F6          DCD5  ;SHOW  -- Display the LSD of VØ
  F8          7C05  ;VC+5  -- Add 5 to X coordinate
  FA          12EE   STAC1 -- Go loop until done

      ;RESET STACK

  FC  STAC2:  8AE0  ;VA=VE -- Restore VA from saved value in VE
  FE          00EE  ;RETN  -- Return -- stack shown on display

      ;MACHINE LANGUAGE SCROLL SUB

0400  9B BE BF F8 00 AE F8 08 AF 4F 5E 1E 8F 3A 09 F8
0410  00 5E 1E 8E 3A 0F D4 00 00 00 00 00 00 00 00 00
```

ANSWERS TO LAST MONTH'S LITTLE LOOPS:

1) The best place to enter a timing loop, slowing the CHIP-8 prime number program to a respectable rate is <u>not</u> in the SHOW PRIME (SHOPR) sub. The purpose of this question was to inspire you to dig into the listing. At address 0264, and only there, we can be sure that V4 is prime. Entering a loop at that point will delay showing each prime number by as much as you want. The following will do the job:

```
0264            1270 -- Jump to patch at 0270
 --
0270            6F60 -- Set VF= constant timing value
  72  LOOP:     7FFF -- Subtract one from VØ by adding hex FF
  74            3F00 -- When VF=00, skip the next instruction
  76            1272 -- Go to loop, waiting a fixed amount of time
  78            123A -- Go show the prime number
```

NOTE: I purposely did not use the regular timing loop that works by setting and testing the internal CHIP-8 timer. There's more than one way to skin a scorpian! (What?)

2) Simple. To change the amount of scroll, just change the number of times the MLS is called. This is easily accomplished by setting a new loop value into VF with the byte at location 0337. Try a value of $06 or an even larger value like $10 or higher and watch the effect. Less than 6 may produce some odd happenings. Also, unless the top line is totally scrolled away, confusing bits of its lower parts may remain on display.

NEW PRODUCT ANNOUNCEMENT

POWER LINE INTERRUPTER
$139.95/$124.95

Electronic Specialists, Inc.
171 South Main Street
Natick, Mass. 01760
(617) 655-1532

Electronic Specialists has introduced the Power Line Interrupter. Should AC line voltage be disrupted or exceed user selectable limits, the Power Line Interrupter disconnects the power. Front panel controls provide under/over voltage interrupt level selection and power reset. Other features include integral spike/surge suppression and response delay to prevent false interrupts.

Intended for specialized MicroComputer applications, where equipment is subject to periods of unattended operation, the Power Line Interrupter is designed to provide safety and protection.

Connected to the AC line with a standard 3-prong plug, the Power Line Interrupter can accommodate a 15 amp resistive load or a 10 amp inductive load.

Model PI-15-O/U Over and Under-Voltage $139.95
Model PI-15-U Under-Voltage only $124.95

For more information, write to Frank Stifter at the above address.

SIX SONGS FOR VIP SUPERSOUND

by John C. Hanner

The Alley Cat Song

Pin-8 #1 Enter (7 pages) Pin-8 program list with test song. (From RCA VIP Supersound System VP550 instruction manual, appendix B,B-2 and B-3)

Measure Time #2 Check 0259, should be FF (slow 4/4 time)

Break codes #3 Make the following changes:

```
        0270 - 1201 B014 01E0 FE11 01F0 1401 B0FE 1308
        0280 - B014 01B0 FF00
        02F0 - 0404 0909 0F0F 0404 0909 0F0F 0909 0F0F
A Measures
        0300 - 0105 090D 1014 181C 1F23 272B 2E32 363A
        0310 - 3E41 4447 4A4D 4F53 575B 5F63 666A 6E72
        0320 - 0000 and 0000's to 037F

B Measures
        0380 - 0105 090D 1115 191D 2125 292D 3135 393D
        0390 - 4144 4649 4B4E 4F53 5559 5D61 6569 6D71
        03A0 - 0000 and 0000's to 03FF

A Notes 0400 - 0070 6F6D 6C6B 6CAD 606B 6C6D 6ECF 60A0
        0410 - 706F 6D6C 6B6C AD60 6B6C 6D6F D060 A070
        0420 - 6F6D 6C6B 6CAD 606B 6C6D 6ECF 60A0 706F
        0430 - 6D6C 6B6C AD60 6B6C 6D6F D060 6070 72B2
        0440 - 60A0 6070 72B2 60A0 6070 72B2 60A0 A072
        0450 - 706F 6D6B 6AA9 6070 6F6D 6C6B 6CAD 606B
        0460 - 6C6D 6ECF 60A0 706F 6D6C 6B6C AD60 6B6C
        0470 - 6D6F D060 A000 0000 and 0000's to 04FF

B Notes 0500 - 0070 606B 6064 606B 6070 606B 6072 606B
        0510 - 6072 606B 6072 606B 6072 606B 6070 606B
        0520 - 6070 606B 6064 606B 6070 606B 6072 606B
        0530 - 6072 606B 6072 606B 6072 6078 6070 6B64
        0540 - 60A9 7071 D260 A46B 71D2 60A6 6D71 F266
        0550 - 6066 60CB 60B0 AB6B 60A4 ABB0 AB70 606B
        0560 - 6066 606B 6066 606B 6066 606B 6066 606B
        0570 - 60F0 0000 and 0000's to 05FF

D Measures
        0600 - 8181 8181 8181 8181 8F8F 8F8F 8F8F 8F8F
        0610 - 8787 8787 8787 9393 8181 8181 8181 878B
        0620 - 0000 and 0000's to 067F

D Notes 0680 - 003D 1B1F 1F1F 3F3D 3D3E 3A70 0000 003D
        0690 - 3E3B 3F7F 0000 and 0000's to 06FF
```

-end-

02.10.21

Edelweiss

Pin-8 #1 Enter (7 pages) Pin-8 program list with test song. (From RCA VIP Supersound System VP 550 instruction manual, appendix B,B-2 and B-3)

Measure Time #2 at 0259 enter BF (slow 3/4 time)

#3 enter the following:

Break codes
0270 - 1301 B014 01E0 FE13 04B0 1401 C0FF 0000

A Measures
0300 - 0103 0406 0709 0C0D 0E10 1113 1416 191A
0310 - 1820 2325 2628 2A2B 2C2E 2F31 3234 3738

B Measures
0380 - 0102 0304 0506 0708 090A 0B0C 0D0E 0F10
0390 - 1114 171A 1D20 2326 292A 2B2C 2D2E 2F32

A Notes 0400 - 00A6 69D0 AE69 C7A6 6666 6769 CBC9 A669
0410 - D0AE 69C7 A669 696B 6DCE CE50 0020 2969
0420 - 6D6B 69A6 69CE A76E B06E CDC9 A669 D0AE
0430 - 69C7 A669 696B 6DCE AEFF 0000 0000 0000

B Notes 0500 - 00CE CACF C8C6 C8C1 C6CE CACF C8C6 C6CE
0510 - C669 7260 6672 606E 7260 6672 6067 6B74
0520 - 646D 7466 6D75 666D 72CE CCC8 CAC6 C66E
0530 - 7260 CE00 0000 0000 0000 0000 0000 0000

D Measures
0600 - 8181 8181 8181 8181 8181 8181 8181 8181
0610 - 8484 8484 8484 8484 8181 8181 8181 878A

D Notes 0680 - 003E 3E3B 3F3D 3A3F 3F3D 3D3B 303F 3F3F

-end-

Washington and Lee Swing

Pin-8 #1 Enter (7 pages) Pin-8 program list with test song. (From RCA VIP Supersound System VP 550 instruction manual, appendix B,B-2 and B-3)

Measure Time #2 at 0259 enter FF (slow 4/4 time)

#3 enter the following:

Break codes
0270 - 1301 B014 01E0 FF00 0000 0000 0000 0000

A Measures
0300 - 0105 0709 0C10 1214 171B 1D1F 2226 282A
0310 - 2E32 3436 393D 3F41 4448 4A4C 4F52 5354
0320 - 5500 0000 0000 0000 0000 0000 0000 0000

B Measures
0380 - 0105 090D 1115 191D 2125 272A 2E32 3437
0390 - 3A3E 4246 4A4E 5052 5457 595B 5F63 676B
03A0 - 6E72 0000 0000 0000 0000 0000 0000 0000

A Notes 0400 - 0060 6B6D 6FB0 AFAD AF6D 6CAB 606B 6D6F
0410 - B0AF ADAB 6F6E AF60 6F70 71B2 B0AF AD6D
0420 - 6CAB 606B 6D6F B2B1 B2B3 7470 6F6D 6B6B
0430 - 6D6F B0AF ADAF 6D6C AB60 6B6D 6FB0 AFB0
0440 - B171 72AD 606D 6C6D B2B0 AFAD 706D AB70
0450 - 6DAB EDEF F0D0 60FF 0000 0000 0000 0000

B Notes 0500 - 0060 6564 6361 6068 6061 6068 6061 6068
0510 - 6061 6564 636D 6068 606D 606E 606F 6068
0520 - 606F 6869 6BAB A968 60A4 6360 686E 6F60
0530 - 6860 ABAA AB68 606D 70B0 6F6F 706F 6D60
0540 - 6D60 6860 6D60 6860 6D60 6860 706F ADAB
0550 - ADA9 A6A4 6360 A0A6 A6A7 A668 6068 686D
0560 - 6968 6063 6364 6768 6664 6361 60A6 6D68
0570 - 6160 FF00 0000 0000 0000 0000 0000 0000

D Measures
0600 - 8181 8181 8181 8181 8181 8181 8181 8181
0610 - 8585 8585 8585 8585 8181 8181 8181 8589

D Notes 0680 - 003F 1E1E 383D 3D3E 3A70 3F3F 3F3F 3F3F

-end-

And the Band Played On

Pin-8 #1 Enter (7 pages) Pin-8 program list with test song. (From RCA VIP Supersound System VP 550 instruction manual, appendix B,B-2 and B-3)

Measure time #2 at 0259 enter BF (slow 3/4 time)

#3 enter the following:

Break codes
0270 - 1301 B014 01E0 FE13 04B0 1401 B0FE 1308
0280 - B014 08B0 FF00 0000 0000 0000 0000 0000

A Measures
0300 - 0104 070B 0E0F 1011 1417 1A1D 2021 2223
0310 - 2629 2C2F 3235 393A 3D40 4346 494A 4B4C
0320 - 4F00 0000 0000 0000 0000 0000 0000 0000

B Measures
0380 - 0104 070A 0D0E 0F10 1E1F 2021 2425 2627
0390 - 2829 2B2D 3031 3233 3536 373A 3D00 0000

A Notes 0400 - 006B 6F6B 6A6F 6A68 6B20 28A6 282B CAD0
0410 - D060 6066 6A6D 6A68 6668 6A6D 6A68 6668
0420 - C6CF CF60 686A 6B6B 6B6A 6A6A 6969 6968
0430 - 8828 6868 6868 6D20 2FD0 7060 6868 6768
0440 - 6B8A 2866 686A AB2D 2FC8 CACB 6B60 6060

```
B Notes  0500 - 0068 6060 6360 6068 6F60 6F60 60CA C3CA
         0510 - 6F63 606F 6060 6A60 606F 6060 AA63 C8C3
         0520 - C860 6360 C8CA CBCD C1AD 6BAA 646A 6060
         0530 - C1C2 C3A4 60CA CF68 6F6B 6860 6000 0000

D Measures
         0600 - 8181 8181 8181 8181 8181 8181 8181 8181
         0610 - 8484 8484 8484 8484 8181 8181 8181 878A

D Notes  0680 - 003E 3D3D 3B3F 3F37 3F3F 703F 3F00 0000
```

-end-

He's Got the Whole World in His Hands

Pin-8 #1 Enter (7 pages) Pin-8 program list with test song. (From RCA VIP Supersound System VP 550 instruction manual, appendix B,B-2 and B-3)

Measure time #2 at 0259 enter FF (slow 4/4 time)

#3 enter the following:

```
Break codes
         0270 - 1301 B014 01B0 FE12 01B0 1501 B0FF 0000

A Measures
         0300 - 0105 080E 1218 1B21 252A 2E34 383E 4248
         0310 - 4C51 545A 5E64 676D 6F71 7200 0000 0000

B Measures
         0380 - 0102 060A 0E12 161A 1E22 262A 2E32 363A
         0390 - 3F43 474B 4F53 575B 5F63 6700 0000 0000

A Notes  0400 - 0060 7070 6DB0 6D69 3072 7030 302D 6F6F
         0410 - 2B88 2F70 6F2D 2D2F B06D 6930 7270 3030
         0420 - 2D70 706F 6BA9 2030 302D 7070 2D89 3072
         0430 - 7030 302D 6F6F 2B88 2F70 6F2D 2D2F 7070
         0440 - 2D89 3072 7030 302D 7070 6F6B A920 3030
         0450 - 2DB0 2D89 3072 7030 302D 6F6F 2B88 2F70
         0460 - 6F2D 2D2F B02D 8930 7270 3030 2DB0 B0AF
         0470 - ABE9 C960 0000 0000 0000 0000 0000 0006

B Notes  0500 - 00E0 6660 6160 6660 6160 6860 6160 6860
         0510 - 6160 6660 6160 6660 6161 6860 6160 6661
         0520 - 6660 6660 6160 6660 6160 6860 6160 6860
         0530 - 6160 6660 6160 6660 6160 6860 6160 6661
         0540 - 6166 6066 6061 6066 6061 6068 6061 6068
         0550 - 6061 6066 6061 6066 6061 6060 6160 6160
         0560 - 6160 6166 6061 6066 6166 6000 0000 0000

D Measures
         0600 - 8181 8181 8181 8181 8181 8181 8181 8181
         0610 - 8181 8181 8181 8182 847F 7F7F 7F7F 7F7F

D Notes  0680 - 007D 7E77 7A00 0000 0000 0000 0000 0000
```

-end-

02.10.24

Pin-8 #1 Enter (7 pages) Pin-8 program list with test song. (From RCA VIP Supersound System VP 550 instruction manual, appendix B,B-2 and B-3)

Measure Time #2 at 0259 enter FF (slow 4/4 time)

#3 enter the following:

Break codes
0270 - 1300 0115 08B0 FE12 01B0 1401 B0FF 0000

A Measures
0300 - 0109 0109 1119 0109 2129 3131 3700 0000

B Measures
0380 - 0109 0109 1119 0109 2129 3131 3500 0000

A Notes 0400 - 0024 3028 342B 372D 392E 3A2D 392B 3728
0410 - 3429 352D 3930 3032 3233 3332 3230 302D
0420 - 2D26 3228 3429 352A 362B 3729 3528 3426
0430 - 3264 2668 2426 28E4 0000 0000 0000 0000

B Notes 0500 - 0024 2028 202B 202D 202E 202D 202B 2028
0510 - 2029 202D 2030 2032 2033 2032 2030 202D
0520 - 2026 2028 2029 202A 202B 2029 2028 2026
0530 - 2064 6068 60E4 0000 0000 0000 0000 0000

D Measures
0600 - 8181 8181 8181 8181 8181 8787 8800 0000

D Notes 0680 - 003E 1B1B 3D1B 1B7F 7700 0000 0000 0000

EDITOR'S NOTE: Because we do not have a Supersound board here, the above code has not been tested. It was retyped with eagle eyes on the author's hand-written copy and we can only cross our fingers and hope it's all correct. If you detect any errors, please let us know so we may include them in a future issue. Thanks. - Tom

PROGRAMMING HINTS

The other day I happened to wave a Simple Sound board in front of my monitor display. To my surprise, the display began to dance and sway as though I were waving a baton to direct the motion! Of course the cause was the large magnet attached to the board's built-in speaker. This was good for a few laughs but I do not know if any harmful effects are possible on the insides of the video "box." Don't try this unless you know the correct answer!

Then I began thinking. A magnet that powerful could affect information recorded on cassettes. This is a serious danger! If you have a Simple Sound board, don't ever, even by accident, set it down on a recorded cassette. You could lose the whole tape.

A HARDWARE BREAKPOINT SYSTEM FOR THE COSMAC VIP

by Bill Barrett

Haven't you wanted to try your hand at writing machine language programs on your VIP, but were frustrated at being unable to figure out why they didn't work? One tiny mistake and the kit just sits there with a blank screen. Of course, you can go into the operating system and poke through the memory locations that hold some of the registers, then try to figure out what went wrong, but after some of that, you are probably ready to be picked up by the men in white jackets with a butterfly net.

Now there's a better way. For about $20 in integrated circuits, and a PC board that you can wire yourself (or order from me), you can make a hardware breakpoint board. That will give you an immensely powerful tool for finding out just what is going on in the program you wrote. You can even single-step through each of the machine instructions, until you see where the trouble begins!

The board

You will need a 44-pin board with enough room for 9 16-pin IC's, and 4 14-pin IC's. Here's a list of the necessary IC's:

(4) 74C175
(4) 4063
(1) 4049
(1) 4081
(1) 4073
(2) 4013
(1) 1N914 silicon signal diode (or equivalent)

The 4063 is made by RCA, but seems to be carried by only a few suppliers -- one such is Tri-Tek, 7808 N. 27th Avenue, Phoenix, AZ, 85021.

A schematic of the board is given in figure 1. The circuit runs on the +5 supply of the VIP and draws very little current. The whole system can run on a minimum-memory Cosmac, although with only 8 pages of memory, your program can only fill a little less than 5 pages.

Here's how it works. The four registers Q1-Q4 carry a sixteen-bit memory address. That address is loaded in two 8-bit portions through a pair of OUT 7 instructions. Once the address is loaded, the four comparators Q5-Q8 continuously compare that 16-bit number with the values on the MA bus. Some additional logic makes sure that the only equality comparison that is effective is when an instruction address exactly matches the register value. When that match is detected, the INTERRUPT line to the 1802 microprocessor is pulled to ground, causing an interrupt cycle.

At that point, a software program (given later) goes into effect. This program is similar to the one I reported in the VIPER, Oct. 1978, but with some upgrades. This program saves all the registers,

You can order a kit containing a high-quality gold-plated circuit board, a cassette tape with the debug software, and complete instructions from: WILLAVON, 1164 Hyde Ave., San Jose, CA 95129. The kit costs $25, including postage and handling. Add 6 1/2% sales tax if you are a California resident. The instructions include complete assembly instructions, a complete source listing of the software, and diagnostic instructions in case of trouble. It does NOT include the IC's listed above -- I suggest ordering them separately from a reliable supplier.

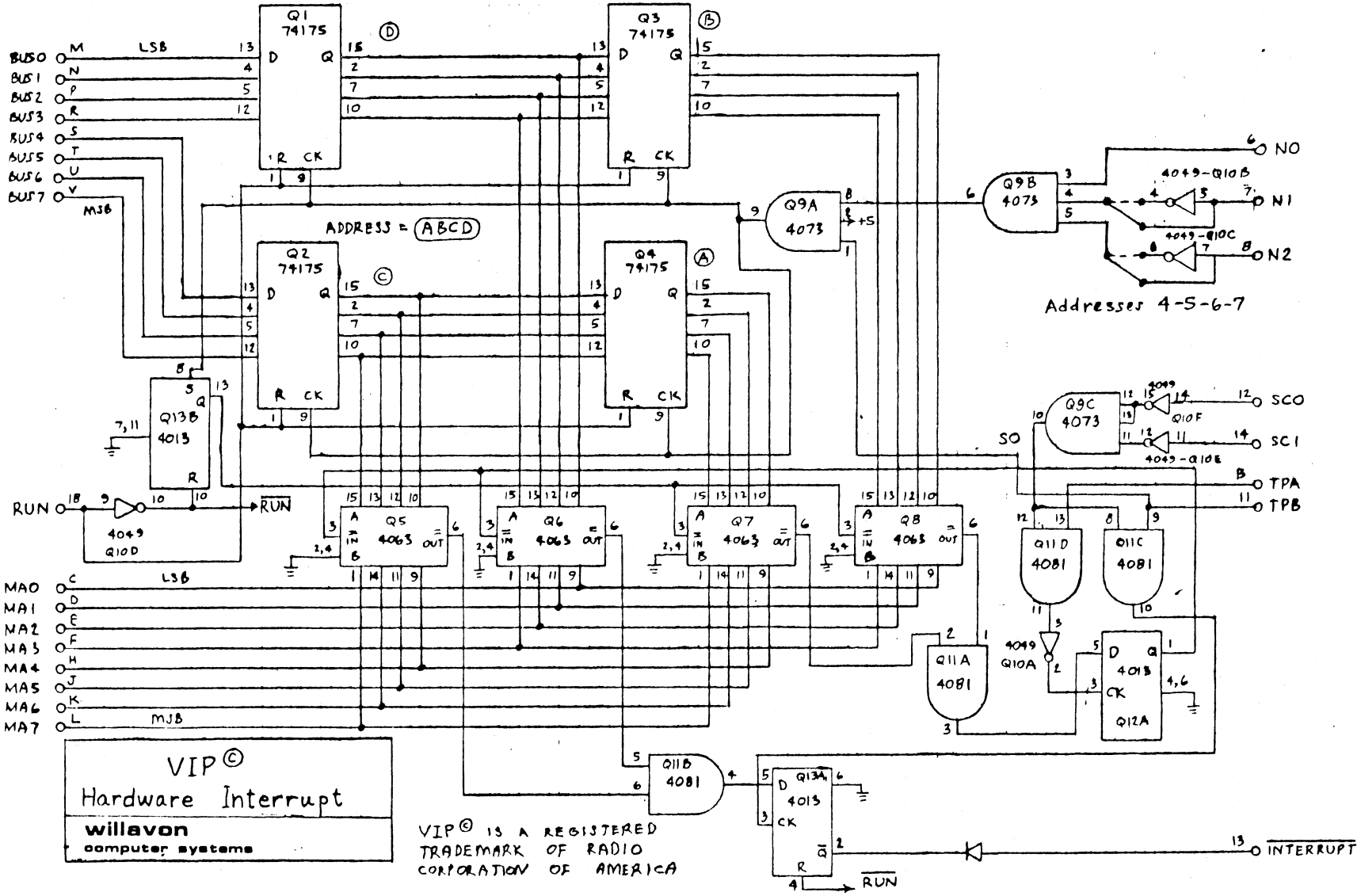


Figure 1. Schematic

displays them on the TV screen, permits you to set a breakpoint at a different location, single-step for a while, view memory or change memory. You can also change almost any of the registers by changing their memory image.

The program operations are shown in figure 2. To use the system, you must have the debug software program loaded in pages N-2 and N-1, where page N is the top page of your memory. For example, for an 8-page (minimum) system, load the program at 0500 for 2 pages. It needs a stack, controlled by register R(2), and that will start at the top of page N-3. Page N is used for the TV display, shared with the VIP operating system. Finally, you need to write C00500 in location 0. That transfers control to an initialization section of the debug software. It will set up the stack and interrupt register, set a breakpoint at location 0003, then return to 3. A breakpoint there will immediately occur so that you can set a breakpoint somewhere else, single-step, or whatever. The registers will be set as follows: X=2, P=3, Q=0, IE=1, R1=055B (the interrupt entry location), R2=04FF (the stack top),R3=program counter, RB.HIGH =memory top page, i.e. 07 for an 8-page system. The TV interface circuit is OFF. The other registers are undefined.

These conditions are somewhat different than those after the operating system starts a program, but I find them more convenient for program development.

The system depends on IE=1, R1 holding 055B and R2 pointing to a valid stack. The TV interface should also be OFF, since otherwise it will cause interrupts. Of course, your program should not alter the instructions in the debug software pages. If any of this goes wrong, the debug system will fail.

The debug program does not alter itself, so it could be embedded in a ROM or in write-protected memory. That goes a long way toward averting special problems with cantankerous programs.

No changes to the debug program are needed for different memory sizes. However, it must be loaded at N-2 as explained above, and the VIP operating system must be in place -- the debug display uses some of the OS features.

If you never invoke the first breakpoint through the initial C00500, the debug board cannot interrupt the VIP, even though you happen to hit an instruction whose address matches the board's address register. The board is deactivated on RESET, and remains that way until an OUT 7 is executed.

You must be careful to plant a breakpoint on the first byte of an instruction, not its second or third byte. That's the only address the debug board will pay attention to.

When a breakpoint is hit, the TV screen will display these registers:

```
X,P,D    Q,DF,Q,DF
R0       R1
R2       R3
R4       R5
R6       R7
```

Touch key 1 and you will see the rest of the registers:

R8	R9
RA	RB
RC	RD
RE	RF

Key 1 will cycle back and forth between these displays. Now suppose you have set a breakpoint on an instruction at location L. When the display appears, that instruction will have been executed, and the program counter R(P) will be pointing to the next one to be executed. If you hit key B, the debug software will automatically set a breakpoint at location R(P) and resume -- one more instruction will be executed and the display will immediately come back. If you hit key F, the current breakpoint position is unchanged, and the program will resume until the instruction at location L is again executed. A new breakpoint location can be set using key C.

The contents of memory can be viewed by using key A and changed by using key 0 -- refer to figure 2 for the method.

The contents of any of the registers except R1 and R2 can be changed by finding their stored location in memory. Here's how: Note the value of R(2) displayed on the screen -- let it be N. Then count backward starting with N-1 for the X-P byte, N-2 for the D byte, etc., until you come to the register bytes you want. Then use key 0 with that address to change the register's value. When you return to the screen display with key 1, you will see the new values. R1 and R2 cannot be changed in this way -- you can change their stored values, but it won't make any difference when you resume the program.

The debug program

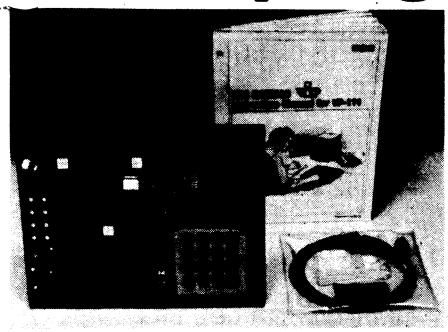
Here's the two-page debug program that makes it all work:

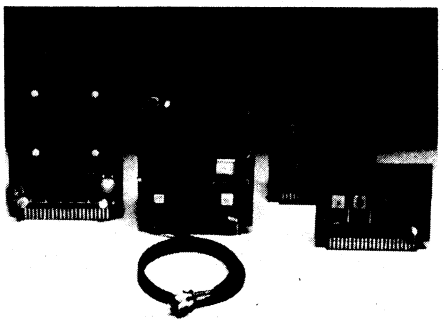
```
0500  91 BB 90 B3 B1 FF 01 B2
0508  81 A2 F8 0F A3 E2 D3 F8
0510  5B A1 22 F8 03 73 F8 00
0518  52 67 67 C0 00 03 E3 F8
0520  24 A1 71 21 61 82 A0 92
0528  B0 20 F8 D1 50 20 F8 AF
0530  50 FE 72 D0 20 20 E0 F0
0538  FB B3 32 49 F0 E2 3B 45
0540  FC 10 50 30 32 FF 11 30
0548  42 E2 60 60 60 60 72 A0
0550  72 B0 60 72 F6 7A 32 59
0558  7B 72 70 22 78 22 E2 73
0560  F8 00 39 66 F8 08 7E 73
0568  73 90 73 80 73 91 73 F8
0570  5B 73 82 FC 09 A0 92 7C
0578  00 73 80 73 82 FF 1A A0
0580  92 7F 00 B0 F8 D1 50 20
0588  F8 93 50 FE FE D0 73 20
0590  20 E0 F0 E2 33 9F FC F0
0598  50 FB A0 32 A3 30 8D FF
05A0  EF 30 98 A8 F8 00 A3 91

05A8  FC 01 B3 D3 DC DC DC 59
05B0  F8 03 AE D4 12 12 12 12
05B8  D6 3A CA 22 17 17 17 19
05C0  99 73 89 73 30 AC 12 12
05C8  12 12 C0 06 21 42 70 22
05D0  78 22 52 C4 C4 C4 9B B0
05D8  F8 00 A0 80 E2 E2 20 A0
05E0  E2 20 A0 E2 20 A0 3C DB
05E8  7A 88 32 CD 7B 28 30 CD
05F0  00 00 00 00 00 00 00 00
05F8  00 00 00 00 00 00 00 00
0600  F8 CF A1 91 FC 02 BB F8
0608  CC A4 91 FC 01 B4 F8 BF
0610  AC 91 FC 01 EC F8 95 A6
0618  F8 81 B6 B5 BA 69 E3 70
0620  23 82 FC 24 A7 92 7C 00
0628  B7 E2 F8 24 A9 89 FF 14
0630  3B 36 F8 14 30 37 89 AE
0638  D4 D6 FB 01 3A 46 89 FF
0640  14 A9 3B 21 30 2D 02 32
0648  93 FF 0F C2 05 1E FF FB

0650  32 93 FF 02 32 79 FF FF
0658  3A 39 82 FC 24 A7 92 7C
0660  00 B7 07 FA 0F FE FC 04
0668  52 87 F7 A7 97 7F 00 B7
0670  E7 67 27 27 67 E2 C0 05
0678  1E E2 82 A7 92 B7 DC DC
0680  DC 60 67 22 22 DC DC DC
0688  60 67 22 F8 02 AE D4 12
0690  D6 30 21 E2 82 A7 92 B7
0698  22 DC DC DC B9 DC DC DC
06A0  A9 07 27 C2 05 AC E2 09
06A8  73 F8 03 AE D4 D6 FB 01
06B0  C2 05 C6 12 19 17 17 89
06B8  57 17 99 57 30 A6 D3 D6
06C0  FE FE FE FE AE D6 8E F1
06C8  73 30 BE D3 9B BD F8 00
06D0  AA F8 FF AD ED 8A 73 8D
06D8  3A D5 5D E7 F8 C6 A5 F0
06E0  F6 F6 F6 F6 D5 F0 27
06E8  0F D5 2E 8E 32 CB 8D FA
06F0  07 3A DF 8D FC 28 AD 30
06F8  DF
```

The VIP hobby computer: Start programming for only $99.





New! VP 111 Microcomputer.... $99. Assembled* and tested.

Features:
- RCA 1802 Microprocessor.
- 1K Bytes static RAM.

Expandable on-board to 4K.
Expandable to 32K Bytes total.
- 512 Byte ROM operating system.
- CHIP-8 interpretive language or machine language programmable.
- Hexidecimal keypad.
- Audio tone generator.
- Single 5-volt operation.
- Video output to monitor or modulator. Cassette interface—100 Bytes/sec.
- Instruction Manual with 5 video game listings, schematics, CHIP-8, much more!

Ideal for low-cost control applications.

Expandable to full VIP capability with VP-114 Kit.

*User need only connect cables (included), a 5-volt power supply, and speaker.

New low price! The original VIP.. $199. Completely assembled and tested.

All the features of the VP-111 plus:
- A total of 2K Bytes static RAM.
- Power supply.
- 8 Bit input port.
- 8 Bit output port.
- I/O port connector.
- System expansion connector.
- Built-in speaker.
- Plastic cover.

Three comprehensive manuals:
- VIP Instruction Manual—20 video game listings, schematics, much more.
- VIP User's Guide—operating instructions and CHIP-8 for the beginner.
- RCA 1802 User's Manual (MPM-201B) - complete 1802 reference guide.

RCA

COSMAC VIP lets you add computer power a board at a time.

With easy-to-buy options, the versatile RCA COSMAC VIP means even more excitement. More challenges in graphics, games and control functions. For everyone, from youngster to serious hobbyist.

Built around an RCA COSMAC microprocessor, the VIP is easy to program and operate. Powerful CHIP-8 interpretive language gets you into programming the first evening. Complete documentation provided.

Send the coupon now...
Complete the coupon below and mail to: RCA VIP Customer Service, New Holland Avenue, Lancaster, PA 17604.

Or call toll free (800) 233-0094
to place your Master Charge or VISA credit card order. In Pennsylvania, call (717) 397-7661, extension 3179.

Please send me the RCA COSMAC VIP items indicated.

- ☐ **VP-111** New low cost Microcomputer (See description above) **$99**
- ☐ **VP-114** Expansion Kit for VP-111—Includes 3K RAM, I/O Port and connectors... **$ 76**
- ☐ **VP-711** VIP—The original VIP Microcomputer (See description above) **$199**
- ☐ **VP-44** RAM On-Board Expansion Kit—Four 2114 RAM IC's. Expands VP 711 memory to 4K bytes ..... **$ 36**
- ☐ **VP-590** VIP Color Board—Converts VIP to color. Four background and eight foreground colors..... **$ 69**
- ☐ **VP-595** VIP Simple Sound Board—Provides 256 programmable frequencies. For simple music or sound effects. Includes speaker ..... **$ 30**
- ☐ **VP-550** VIP Super Sound Board—Turns your VIP into a music synthesizer! Two independent sound channels. On-board tempo control. Outputs to audio system..... **$ 49**
- ☐ **VP-551** 4-Channel Super Sound—Includes VP-576 expander, demo cassette and manual. Requires VP-550 and 4K RAM ..... **$ 74**
- ☐ **VP-570** VIP Memory Expansion Board—Plug-in 4K RAM memory ..... **$ 95**
- ☐ **VP-580** VIP Auxiliary Keypad—Adds two-player interactive capability. 16-key keypad with cable. Connects to sockets on VP-590 or VP-585..... **$ 20**
- ☐ **VP-585** VIP Keypad Interface Board—Interfaces two VP-580 Auxiliary Keypads to VIP. ..... **$ 15**
- ☐ **VP-560** VIP EPROM Board—Interfaces two 2716 EPROMs to VIP..... **$ 34**
- ☐ **VP-565** VIP EPROM Programmer Board—Programs 2716 EPROMs With software ..... **$ 99**
- ☐ **VP-575** VIP Expansion Board—Provides 4 buffered and one unbuffered expansion sockets..... **$ 59**
- ☐ **VP-576** VIP Two-Board Expander—Allows use of 2 Accessory Boards in either I/O or Expansion Socket..... **$ 20**
- ☐ **VP-601** ASCII Keyboard--128-character ASCII Encoded alphanumeric keyboard ..... **$ 65**
- ☐ **VP-611** ASCII/Numeric Keyboard—Same as VP-601 plus 16 key numeric keypad..... **$ 80**
- ☐ **VP-620** Cable: Connects ASCII keyboards to VIP ..... **$ 20**
- ☐ **VP-700** VIP Tiny BASIC ROM Board—BASIC code stored in 4K of ROM..... **$ 39**
- ☐ **VP-710** VIP Game Manual—Listing for 16 exciting games..... **$ 10**
- ☐ **VP-720** VIP Game Manual-II—More exciting games (Available 2nd qtr. '80) ..... **$ 10**
- ☐ **MPM-201B** CDP1802 User Manual—(Included with VP-711) ..... **$ 5**
- ☐ Please send more information..... —

Enclosed is $_____ for items checked plus shipping & handling charge of $3.00.
Add your state and local taxes $_____ Total enclosed $ _____
I enclose ☐ check or ☐ money order or, charge my ☐ VISA/Bank Americard
☐ Master Charge.

Credit card account No. _____
Master Charge Interbank No. _____ Expiration date: _____

Signature (required for credit card orders): _____
Name (please type or print): _____
Street address: _____ City: _____
State & Zip: _____ Telephone: ( ) _____

Make checks payable to RCA Corp. Prices and specifications are subject to change without notice.

02.10 . 30

9600 BAUD I/O WITH THE VIP

by Bill Barrett

Yes, that's right. The 1802 can interface with a CRT at 9600 baud with a couple of simple software subroutines. That's about 900 characters per second!

You'll need a couple of IC's -- two line drivers and a line receiver. One line driver connects to the Q output of the VIP. Another should be connected to a pin of U24, the byte output latch. The line receiver should go to pin X of the byte I/O Interface.

On the terminal side, there are only four RS232 connections to worry about. These are:

Pin 12, BA, data out: goes to your line receiver. This will carry characters from the terminal to the 1802.

Pin 48, ground. Nuff said.

Pin 42, BB, data in: this goes to the line driver connected to Q. We will toggle Q to send a character to the terminal.

Pin 44, CB, clear to send: this goes to the other line driver. This line should go high (+3 volts) to inform the terminal that the 1802 wants to send characters.

The two data lines will normally be LOW (-3 volts) when no information is being transmitted. One eight-bit character is transmitted (either way) by sending binary levels. Suppose the character is

c7 c6 c5 c4 c3 c2 c1 c0

in binary, where c7 is the most significant bit and c0 the least. Then the levels sent are:

1, to indicate a 'start'
c0, i.e. NOT c0 -- if c0=1, we send 0 and vice versa.
c1
c2
...
c7
0, the first 'stop'
0, the second 'stop'

A character like that can be sent at any time -- the terminal waits for the leading edge of a 'start' pulse, then begins timing out the center positions of each of the remaining levels. The time spent for each bit must be 1/B seconds, where B is the baud rate. For 9600 baud, that's 104.2 microseconds on each level. In the 1802, each 2-cycle instruction takes 9.088 microseconds, so 11.5 such instructions (10 2-cycle and 1 3-cycle instruction) can be

executed for each level to get the timing right.

The software needed to write one character is given below, and is called WRCHAR. It is written on the assumption that it will be called by a SEP n, where n is any register set to the address of WRCHAR. However, this can also be called by the standard subroutine call and return technique (SCRT) by replacing the final branch by RETN (SEP5). It expects D to contain the character to be transmitted. (Note that the RCA SCRT does not preserve D.) It uses F as a temporary. It assumes that when Q is SET, the transmitted level to the terminal is HIGH.

```
(WRCHAR writes one character
   to terminal)
   (Works at 9600 baud)
WRCHARRET:
 SEP  R3; (Fix this if
   another P was used for
   the call)
WRCHAR: (Any program
   register can be used.
   Character must be in D at
   this point)
 PHI  RF;
 SEQ; (start, goes out
   through Q)
 LDI  8;
 PLO  RF; (bit counter)
 NOP; NOP; NOP; (7.5
   instruction times)
W3:
 GHI  RF;
 SHR; (bit to DF)
 PHI  RF;
 BDF  W1;
 SEQ;
 BR   W2;
W1:
 REQ;
 BR  W2;
W2:
 NOP; (for timing)
 DEC  RF;
 GLO  RF;
 GLO  RF; (for timing)
 BNZ  W3;
 NOP; NOP; NOP; (last pulse)
 REQ;
 LDI  10;
W6:
 SMI  1;
 BNZ  W6; (Final stop pulses)
 BR   WRCHARRET;
```

Reading a Character

One character is read from the terminal by calling subroutine READCHAR given below. This can be called whenever input from the terminal is expected by the VIP. It 'hangs' on the BN4 instruction at L1 until the first HIGH is sensed on the data out line from the terminal, then starts its timing countdown, picking up bits and packing them into a byte. At the end, it waits out two periods, then returns with the received byte in the D register. This assumes that a HIGH level from the terminal results in line E4 going HIGH.

Again, this subroutine can be called through SCRT or through a SEP n. It uses register F as a temporary.

```
(READCHAR reads one character
   from terminal)
   (Operates at 9600 baud)
READCHARRET:
 SEP  R3;
READCHAR: (PC=5)
L1:
 BN4  L1; (Wait for space)
 LDI  8;
 PLO  RF;
 LDI  6;
L3:
 SMI  1;
 BNZ  L3; (Wait for middle of
   second pulse)
L5:
 NOP; (For timing)
L7:
 B4   L2;
 LDI  1;
 BR   L4;
L2:
 LDI  0;
 BR   L4;
L4:
 SHR;
 GHI  RF;
 SHRC;
 PHI  RF;
 DEC  RF;
 GLO  RF;
 BNZ  L5;
L6:
 B4   L6; (Wait for final mark)
 GHI  F; (Returned in D and in
   F.HIGH)
 BR   READCHARRET;
```

Parity

If you have to supply a parity bit as part of the information, I suggest setting that up before calling WRCHAR. There isn't much surplus time to compute parity on the fly. Also DF is used in the bit writing loop, so it can't easily be used for parity purposes. Parity is really only necessary if you have a noisy channel between the VIP and the terminal. A hard-wired logic channel requires no such precautions.

Terminal-VIP Handshaking

From here on, you will have to find out how your terminal expects to communicate with a computer system. There are several different kinds of handshaking possibilities. Some of them make use of additional RS232 connections, such as the request-to-send (pin 13) and the clear-to-send (pin 44). My terminal (an HP2621) accepts a wide variety of handshaking methods, and is easily changed from one to another. Yours may want just one way of communicating.

The simplest form of I/O is probably line-oriented ENQ-ACK -- that's the one I chose for my 2621. My terminal has a built-in buffer of 125 character maximum. It can accept characters at full rate continuously if only the CRT display is on. However, my attached printer is rather slow, so when using a printer, the VIP must stop every so often and wait for the printer to catch up. Here's how the system works:

Before the VIP sends a batch of characters (less than the 125 character buffer capacity), it sends a ENQ character. When the terminal's buffer is empty, the terminal sends an ACK character. The VIP of course is waiting for that character. When it comes

out, the VIP starts sending characters to the terminal. No more than 125 are sent -- usually just one line, then another ENQ is sent, etc.

While the VIP is sending characters, it can be oblivious to anything returned from the terminal, or it can notice whether a start pulse is on its receive line. There's enough time to get into the receive subroutine to look at the character if you want to do that. I don't bother.

Eventually, the VIP tires of sending information to the terminal, and goes into a receive mode by calling RCHAR. Here the VIP just waits until something appears on its EF4 line, then picks it up. If it then goes into some long processing cycle, it won't notice anything else sent to it until it's done. That's usually OK. You can probably arrange for the cursor on the terminal to blank out in the meantime, to indicate that the VIP isn't listening for a while. On my terminal, the clear-to-send line serves that purpose -- when HIGH, the terminal is in a send state, with cursor and keyboard active, and when LOW, it is in a receive state.

Dear VIPERS-

This is the very last issue of the VIPER from Aresco. As you will note elsewhere, Don Hartley will be publishing the next volume.

I'll really miss you. Through your phone calls and letters, I've come to know most of you reasonably well - and it hurts to say good-bye.

I'll send all your names to Don so he can let you know when he's ready for your renewals. Please give him the same friendly support you've given me - the VIPER can't continue at all without you.

I'm probably going to be totally blind in less than a year. I have known only since my trip to California (the trip that delayed the last issue). Although I was able to publish last year, it seems to me to be unethical to try for another year when I may not be able to see at all by January.

Many of you have called and asked why I haven't sent renewal notices; you've sent letters referring me to agencies and products which may prove to be of immense value. Your affection has helped a great deal - and I love you all for it. (Some of you still didn't know that I'm a woman after two years of VIPER! I really enjoyed your astonishment.)

So good-bye, friends. Give Don all the help you can, because you have made VIPER what it is: people sharing with other people.

Regards,

Jerry